RPM Configuration File Handling Mechanism

Techknow Grid	
Language	Linux
Platform	Linux
Level	Intermediate

— Raja R. K

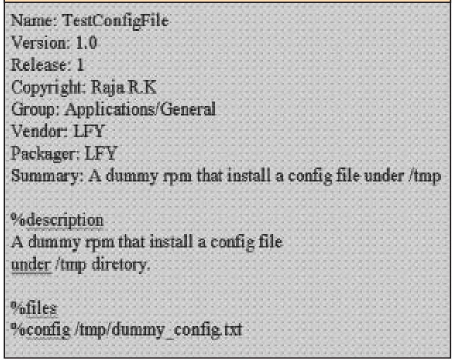
In this article the author demonstrates how the RPM tool handles your customized configuration file under different scenarios

As a System Administrator or Linux user, you may have spent several hours in customizing your system configuration files that are installed as part of Linux. You do not want them to be updated or deleted for any reason when you reinstall or upgrade the packages that installed those configuration files. In the case of RedHat or Fedora based distribution, RPM (an acronym for RedHat Package Manager) can be used to reinstall or upgrade packages on your system.

RPM is a widely used software package management tool in Linux. This powerful and easy-to-use tool has the capability to install, uninstall, reinstall, upgrade, delete and verify software packages. Please see the reference section to know more about RPM.

RPM's configuration file handling mechanism could be best explained with a simple example. Consider the RPM spec file shown in **figure 1**.

Fig 1 A sample RPM spec file

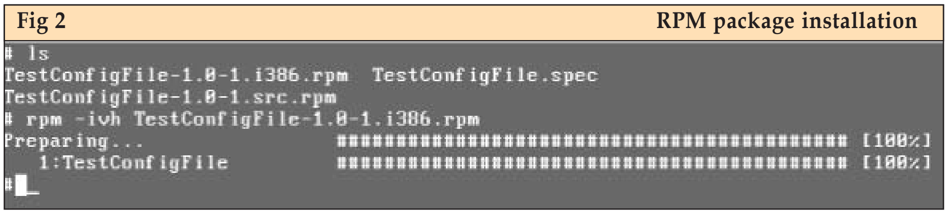


```
Name: TestConfigFile
Version: 1.0
Release: 1
Copyright: Raja R.K
Group: Applications/General
Vendor: LFY
Packager: LFY
Summary: A dummy rpm that install a config file under /tmp

%description
A dummy rpm that install a config file
under /tmp diretory.

%files
%config /tmp/dummy_config.txt
```

The section of the RPM spec file that we are interested in is the %config directive under %files section. This instructs the RPM tool that the file specified (in the above case /tmp/dummy_config.txt) is a configuration file and has to be treated with special care during installation, upgrading or delete operations.

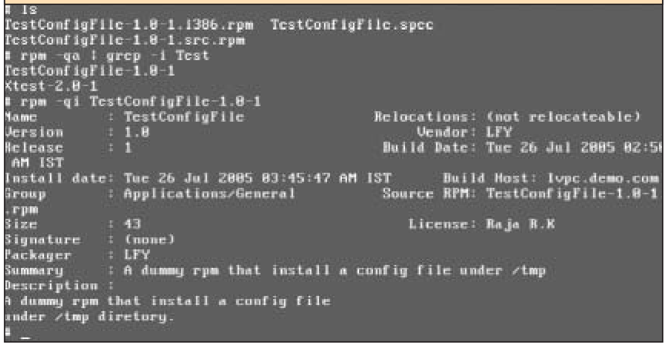
Build the sample RPM spec file using the rpmbuild command and install it as shown in **figure 2**. Prior to installation, be sure to remove the file dummy_config.txt from /tmp directory.

Fig 2 RPM package installation



```
# ls
TestConfigFile-1.0-1.i386.rpm  TestConfigFile.spec
TestConfigFile-1.0-1.src.rpm
# rpm -ivh TestConfigFile-1.0-1.i386.rpm
Preparing...                ########################################### [100%]
   1:TestConfigFile         ########################################### [100%]
#
```

Fig 3 Package Query with RPM

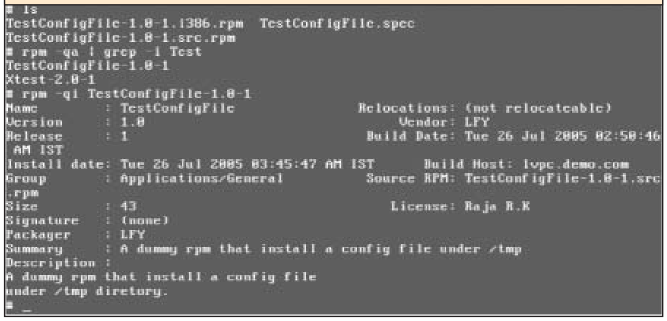


```
# ls
TestConfigFile-1.0-1.i386.rpm  TestConfigFile.spec
TestConfigFile-1.0-1.src.rpm
# rpm -qa | grep -i Test
TestConfigFile-1.0-1
Xtest-2.0-1
# rpm -qi TestConfigFile-1.0-1
Name        : TestConfigFile                Relocations: (not relocateable)
Version     : 1.0                                Vendor: LFY
Release     : 1                              Build Date: Tue 26 Jul 2005 02:5
 AM IST
Install date: Tue 26 Jul 2005 03:45:47 AM IST      Build Host: lvpc.demo.com
Group       : Applications/General           Source RPM: TestConfigFile-1.0-1
.rpm
Size        : 43                                License: Raja R.K
Signature   : (none)
Packager    : LFY
Summary     : A dummy rpm that install a config file under /tmp
Description :
A dummy rpm that install a config file
under /tmp diretury.
# _
```

Verify the installation, as shown in **figure 3**.

Use -qc option to query the list of configuration files set up as part of the package. Refer **figure 4**.

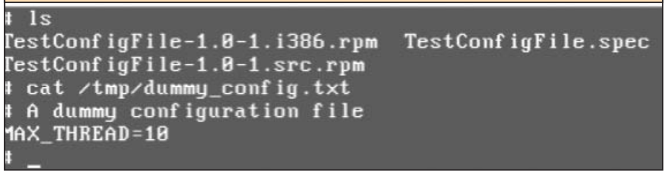
Fig 4 Configuration File Query with RPM



```
# ls
TestConfigFile-1.0-1.i386.rpm  TestConfigFile.spec
TestConfigFile-1.0-1.src.rpm
# rpm -qa | grep -i Test
TestConfigFile-1.0-1
Xtest-2.0-1
# rpm -qi TestConfigFile-1.0-1
Name        : TestConfigFile                Relocations: (not relocateable)
Version     : 1.0                                Vendor: LFY
Release     : 1                              Build Date: Tue 26 Jul 2005 02:50:46
 AM IST
Install date: Tue 26 Jul 2005 03:45:47 AM IST      Build Host: lvpc.demo.com
Group       : Applications/General           Source RPM: TestConfigFile-1.0-1.src
.rpm
Size        : 43                                License: Raja R.K
Signature   : (none)
Packager    : LFY
Summary     : A dummy rpm that install a config file under /tmp
Description :
A dummy rpm that install a config file
under /tmp diretury.
# _
```

From figure 4, you can see that the RPM has correctly recognized our /tmp/dummy_config.txt as Configuration File. **Figure 5** shows the content of the dummy configuration file.

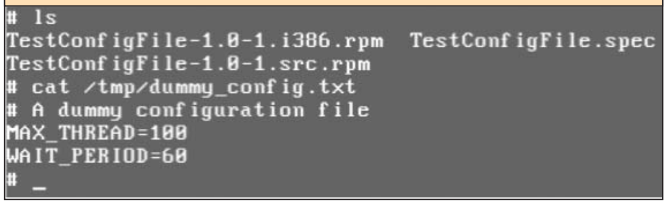
Fig 5 Dummy Configuration File Content



```
# ls
TestConfigFile-1.0-1.i386.rpm  TestConfigFile.spec
TestConfigFile-1.0-1.src.rpm
# cat /tmp/dummy_config.txt
# A dummy configuration file
MAX_THREAD=10
# _
```

Our dummy configuration file has only one parameter (MAX_THREAD) to configure. Now let us customize our configuration file to the one in **figure 6**.

Fig 6 Customized Dummy Configuration File

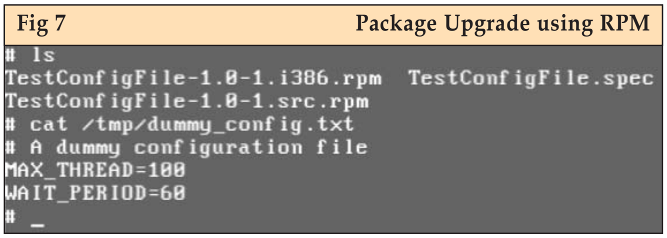


```
# ls
TestConfigFile-1.0-1.i386.rpm  TestConfigFile.spec
TestConfigFile-1.0-1.src.rpm
# cat /tmp/dummy_config.txt
# A dummy configuration file
MAX_THREAD=100
WAIT_PERIOD=60
# _
```

By looking at figure 6, you will observe that we modified the MAX_THREAD parameter to 100 and added a new parameter WAIT_PERIOD. Since we have customized our configuration file, we do not want the RPM package to update or delete this for any reason during reinstallation or upgrading.

Let us now see how RPM handles the configuration file during upgrading. To upgrade our package, we have to update at least the package version or release number in the spec file, otherwise the RPM tool will report error when you install the new package. Change the package release to value 2 and rebuild using rpmbuild command. The package name will now be TestConfigFile-1.0-2.i386.rpm. Upgrade the package as shown in **figure 7**.

Fig 7 Package Upgrade using RPM



```
# ls
TestConfigFile-1.0-1.i386.rpm  TestConfigFile.spec
TestConfigFile-1.0-1.src.rpm
# cat /tmp/dummy_config.txt
# A dummy configuration file
MAX_THREAD=100
WAIT_PERIOD=60
# _
```

From figure 7 you can see that the RPM tool recognized that our configuration file at /tmp/dummy_config.txt has been modified and is different from the one shipped along with the new package. So, it took a back-up with .rpmsave as the suffix of the customized file and installed the new one. Now it is up to the user to verify the new and old file and take the appropriate action.

How does the RPM tool achieve this?

In order to make right decisions about the configuration files, the RPM tool uses three MD5 checksums. These are:

1. MD5 checksum of the file originally installed;
2. MD5 checksum of the file at the time of upgrading; and
3. MD5 checksum of the file in the new package.

Following are some of the other scenarios that you may encounter:

- **You have not customized the configuration file but it has been modified in the new package:** In this case, the RPM tool will just install the configuration file from the new package; and
- **You have customized the configuration file but the file in the upgrade package is same as what you originally installed:** In this case the RPM tool will not install the file from the upgrade package. It leaves the existing file (the file with your changes) intact.

From the above examples, it should be evident that the RPM tool will take the best possible approach to save invaluable efforts spent by the user in customizing the configuration file by handling them intelligently. DIQ

Reference:

http://www.rpm.org/
http://www.rpm.org/max-rpm

The author is a Lead Engineer with HCL Technologies, Chennai. He can be contacted at: rajark_hcl@yahoo.co.in.